```lua
local l = require"lib"
local the = {bins=8, ratios=256}
local about,col,data,row = {},{},{},{}

function aka(klass, instance)
  klass.__tostring = l.cat
  klass.__index    = klass
  return setmetatable(instance,klass) end

-------------------------------------------------------------------------------
local _is={
  nom   = "^[a-z]",  -- ratio cols start with uppercase
  goal  = "[!+-]$",  -- !=klass, [+,-]=maximize,minimize
  klass = "!$",       -- klass if "!"
  skip  = "_$",      -- skip if "_"
  less  = "-$"}      -- minimize if "-"

function about.new(sNames)
  local i = aka(about, {names=sNames, all={}, x={}, y={}, klass=nil})
  for at,name in pairs(sNames) do
    local one = i.push(i.all, col.new(name,at))
    if not name:find(_is.skip) then
      l.push(name:find(_is.goal) and i.y or i.x, one)
      if name:find(_is.klass) then i.klass=one end end end
  return i end

function about.add(i,t)
  local row = t.cells and t or row.new(i.about, t)
  for _,cols in pairs{i.x,i.y} do
    for _,coll in pairs(cols) do
      col.add(coll, row.cells[coll.at]) end end
  return row  end

-------------------------------------------------------------------------------
function col.new(txt,at)
  txt = txt or ""
  return aka(col, {
           n    = 0,                  -- how many items seen?
           at   = at or 0,            -- position ot column
           txt  = txt,                -- column header
           isNom= txt:find(_is.nom),
           w    = txt:find(_is.less) and -1 or 1,
           ok   = true,               -- false if some update needed
           _has  = {}}) end           -- place to keep (some) column values.

function col.add(i,x)
  if x ~= "?" then
    i.n = i.n + 1
    if   i.isNom
    then i._has[x] = 1 + (i._has[x] or 0)
    else local pos
         if     #i._has < the.ratios      then pos=  1 + (#i._has)
         elseif l.rand() < the.ratios/i.n then pos=l.rand(#i._has) end
         if pos then
           i.ok=false -- kept items are no longer sorted
           i._has[pos]=x end end end end

function col.mid(i)
  if i.isNom then
    local mode,most=nil,-1
    for k,v in pairs(i._has) do if v>most then mode,most=k,v end end
    return mode
  else
    return l.per(col.has(i),.5) end end

function col.div(i)
  if   i.isNom
  then local e=0
       for _,v in pairs(i._has) do
         if v>0 then e=e-v/i.n*math.log(v/i.n,2) end end
       return e
  else local t=col.has(i)
       return (l.per(t,.9) - l.per(t,.1))/2.56 end end

function col.has(i)
  if i.isNom then return i._has end
  if not col.ok then table.sort(i._has) end
  i.ok=true
  return i._has end

function col.where(i,x,     a,b,lo,hi)
  if i.nom then return x else
    a = has(i)
    lo,hi = a[1], a[#a]
    b = (hi - lo)/the.bins
    return hi==lo and 1 or math.floor(x/b+.5)*b  end end

function col.norm(i,num)
  local a= has(i) -- "a" contains all our numbers,  sorted.
  return a[#a] - a[1] < 1E-9 and 0 or (num-a[1])/(a[#a]-a[1]) end

-------------------------------------------------------------------------------
function row.new(about,t)
  return aka(row,{_about=about, cells=t, cooked=l.map(t,l.same)}) end

function row.better(i,j)
  i.evaled,j.evaled= true,true
  local s1,s2,d,n,x,y=0,0,0,0
  local ys,e = i._about.y,math.exp(1)
  for _,col in pairs(ys) do
    x,y= i.cells[col.at], j.cells[col.at]
    x,y= col.norm(col,x), col.norm(col,y)
    s1 = s1 - e^(col.w * (x-y)/#ys)
    s2 = s2 - e^(col.w * (y-x)/#ys) end
  return s2/#ys < s1/#ys end

-------------------------------------------------------------------------------
local data={}
function data.new(t)
  return aka(data,{rows={}, about=about.new(t) }) end

function data.add(i,t)
  if i then l.push(i.rows, about.add(i.about,t)) else i=data.new(t) end
  return i end

function data.load(sFilename,         i)
  l.csv(sFilename, function(row) i=data.add(i,row) end)
  return i end

function data.mid(i) return l.map(i.about.y, col.mid) end

function data.bins(i)
  for _,col in pairs(i.about.x) do
    for _,row in pairs(i.rows) do
      local x = row.cells[col.at]
      if x~= "?" then
        row.cooked[col.at] = where(col,x) end end end end

-------------------------------------------------------------------------------
d=data.load("../data/auto93.csv")

l.chat(d:mid())
--map(d.about.x, chat)
-- chat(d.about.x)
-- bins(d)
-- for _,row in pairs(d.rows) do l.chat(row.cooked) end
-- for i=1,20 do
--   r1=l.any(d.rows)
--   r2=l.any(d.rows)
-- end
```